DOCUMENT RESUME

ED 102 947 IR 001 626

AUTHOR Bork, Alfred
TITLE Student Computer Dialogs Without Special Purpose Languages.
INSTITUTION California Univ., Irvine. Physics Computer Development Project.
PUB DATE 2 Jan 75
NOTE 14p.

EDRS PRICE MF-$0.76 HC-$1.58 PLUS POSTAGE
DESCRIPTORS *Computer Assisted Instruction; *Computer Programs; *Computers; Man Machine Systems; Physics; *Programing Languages; Time Sharing
IDENTIFIERS FORTRAN; METASYMBOL; Physics Computer Development Project; University of California at Irvine

ABSTRACT

The phrase "student computer dialogs" refers to interactive sessions between the student and the computer. Rather than using programing languages specifically designed for computer assisted instruction (CAI), existing general purpose languages should be emphasized in the future development of student computer dialogs, as the power and flexibility of general purpose languages make their extension more useful to a larger proportion of computer installations. Two programing languages have been used at the University of California at Irvine--METASYMBOL, the assembly language used on the Xerox Sigma 7, and FORTRAN. A number of small modules have been written in these two languages, and more are being developed. Technological advances in hardware may soon lead to stand-along systems for student computer dialogs, thus making small-scale programs more important. An example of one module developed at Irvine is provided. (DGC)

BEST COPY AVAILABLE

ED102947

STUDENT COMPUTER DIALOGS WITHOUT SPECIAL PURPOSE LANGUAGES

Alfred Bork
Physics Computer Development Project
University of California
Irvine, California 92664

(714) 833-6911

U.S. DEPARTMENT OF HEALTH, EDUCATION & WELFARE
NATIONAL INSTITUTE OF EDUCATION
THIS DOCUMENT HAS BEEN REPRODUCED EXACTLY AS RECEIVED FROM THE PERSON OR ORGANIZATION ORIGINATING IT. POINTS OF VIEW OR OPINIONS STATED DO NOT NECESSARILY REPRESENT OFFICIAL NATIONAL INSTITUTE OF EDUCATION POSITION OR POLICY

January 2, 1975

Introduction

By a student computer dialog we mean a "conversation" between a student and a computer device, possibly a terminal or display. In such a dialog the computer is "talking" with the student, either asking questions and waiting for answers, or responding to requests from the student. This type of usage is sometimes referred to as computer-assisted instruction or CAI. But, because of the intrinsically wider connotation of the term, due to the way computers can be used in instruction, we prefer to call such interactions dialogs. The model for a student-computer dialog is what goes on between the student and the instructor in the instructor's office. We cannot hope to obtain this flexibility with current day computer techniques, with sizable numbers of students, but we can, experience shows, approach a reasonably good approximation, particularly if very skilled teachers are responsible for preparing the student-computer dialogs. In areas where the computational skills of the computer are important, we can offer resources not available to the instructor in such office conversations.

Standard Approach

A typical way this problem has been attacked in the past is to develop a special purpose language for preparing such dialogs. Dozens of such languages have been developed, although few have

ED 001 926

BEST COPY AVAILABLE

had wide-scale acceptance. The original language to attain wide usage was Coursewriter, developed by IBM at Yorktown Heights. Coursewriter is still in existence both on IBM computers and on a variety of other systems (such as Hewlett-Packard). Other examples of such special languages are PLANET, PILOT, and DITRAN. Perhaps the most extensive language of this kind ever developed is TUTOR, used as the basis of the PLATO Project at the University of Illinois.

Alternate Approach

An alternate to developing special languages is to work within existing languages and to make extensions to these languages to make them easier to use in preparing dialogs. There is nothing new about this approach--dozens of attempts have taken this or related directions--but this approach has achieved far less publicity than the special purpose languages. Indeed a document I saw recently attempted to define "hardcore CAI" as something that was written in a special purpose language developed just for this purpose. This seems to be an unfortunate attitude. The student running the dialog does not necessarily know anything about the language facility in which the dialog was prepared; many dialogs currently in use could have been written in a wide variety of languages including many of the standard all-purpose languages. So the software involved is not a basis for classification.

Advantages exist in working within available languages. First, these languages exist and survive because they are powerful languages at least for some purposes. The facilities these languages include would, in many cases, need to be present in any

fully developed dialog language, and so would have to be duplicated. Thus, computational facilities such as those in FORTRAN, ALGOL, or PL/1 would very likely be required in dialogs in physics and in other areas of science. Some of the special purpose languages could provide these facilities by allowing "exits" or hooks to general purpose facilities, and many languages allow this. But working directly within general purpose languages gives another way of attaining these necessary facilities.

Another distinct advantage of working with existing languages is that we can mold the facilities required on pedagogical demand, rather than through theoretical arguments on language design or the way things should be taught. Thus, this approach can allow different authors, teachers from different backgrounds, to create quite different kinds of learning materials. Then these programs can compete in the educational marketplace. New facilities can be added to meet author demands. Thus, in our case the addition of graphic capabilities, essential for teaching applications, was possible with reasonable effort.

Furthermore, some of these general purpose languages allow access to all the resources of the computer, while special languages often restrict such resources. Thus, the ability to use overlay facilities may not be possible in such special software, but this facility or one equivalent to it will be important for developing very large dialogs. Further, because there is no "language," no single language, no fixed pedagogical style is imposed on course authors; existing "CAI" languages often were developed with a particular pedagogical approach in

mind, a predetermined view of how to use the computer in learning. A more general set of facilities does not restrict the author with regard to teaching styles but allows individual teachers to proceed in ways that seem reasonable to them.

There are clearly, however, some disadvantages and problems in using general-purpose languages for dialogs. The first serious question is just how we tag the new facilities, needed for an efficient dialog authoring process, onto an existing language. The mechanism for doing this in the available language may be rather clumsy; subroutine calls, for example, are often not flexible. Furthermore, these extensions may not be at all easy to transport from one machine to another, and so the movement of materials to new environments may not be an easy task.

However, these disadvantages will be overcome by developments in hardware and software in the next few years. At the present, I believe, a sufficient amount of good, well-tested material does not exist in most subject areas. By the time such material is available the hardware situation will be different, and so the transport problems will be different.

The Irvine Dialog Approach

As indicated, working with existing powerful languages allows a great number of variants. For the remainder of this paper I describe the variants in software and authoring that we have employed at Irvine in the Physics Computer Development Project.

First, our critical beginning point is with teaching materials, produced by competent teachers. The idea is that software development does not precede the development of learning sequences, but

follows such development. Thus, competent teachers write the dialogs first, in a form which is not language-dependent, and then we develop whatever new software facilities are necessary for getting those dialogs running. The older facilities developed for previous programs will typically still be useful, and many programs will not demand new facilities. But on the other hand, some programs may require extensive new software. This turning about of the usual direction (first languages, then courseware) with special purpose languages serves to emphasize that our primary purpose is the development of teaching materials, not the development of languages.

Furthermore, as indicated above, we wanted to make use of all the facilities on a general purpose, multi-language, timesharing computer. That is, we did not want to restrict ourselves to a subset of the possibilities on the machine.

In view of these objectives we have worked at Irvine in two general purpose languages, producing dialogs which are combinations of these two languages. The first, and perhaps most important, language is METASYMBOL, the assembly language available on the Xerox Sigma 7 we use. The second is the well-known general purpose scientific language, FORTRAN.

The extension mechanism that we use in the assembly language is the macro or, as referred to on the Sigma 7, procedure. Thus, a typical segment of code for a dialog will consist of macro calls. All of these have English language command names, and in general our policy is to have few arguments in the macro. New macros are written only in response to pedagogical demand.

We can get a better idea of what this looks like if I show a fragment of a program, explaining what the statements mean.

I will begin with the dialog as formulated by the authors. The example I use was generated at a workshop at Irvine for University of California physics faculty from other campuses. It was a group activity (involving Richard Ballard, Irvine; Sun Yiu Fung, Riverside; Peter Geisler, Davis; Bruce Rosenblum, Santa Cruz; and Robert Eisberg, Santa Barbara) intended to illustrate the process of creating dialogs. We began with a flowchart, a fragment of which is produced in a much reduced version here.

BEST COPY AVAILABLE

purpose

The proportionality constant depends on the units chosen. In one common set of units, MKSA or System International,

$$F = \frac{1}{4\pi\varepsilon_0}\frac{Q1\,Q2}{r^2}$$

Where charge is in COULOMBS, distance in METERS, and ε_0 = _____.

RESET BAD2
newpoint
delete 1

Next on our agenda is the direction of the force.

Here's a positive charge, A

delay 2

delay 2

and another B also positive

delay 1

delay 2

A + B not on same point - line!

The force on B is a vector quantity. Establish an origin, indicate it by an arrow starting at B. With the cursor point to the tip of the arrow. Type a point on the circle with the cursor

arrow returns to center
+ Draw Order around B
CURSOR

* other cases
Bump BAD2
& GOTO CURSOR POSITION
1 2

* Ok. ± 5°

Looks good

* between 5° & 45° off
Try to be a little more accurate

* Toward A, ± 45°

No. The force is repulsive (like poles) not attractive. Try again

That direction is right. Try again

You're a little cockeyed, but I suppose you could call it in this direction

The force is directed away from A, as shown by the arrow.

delay 2
+ draw arrow to current point on circle

Note that the authors produced only rough sketches of the drawings required. The flowchart then went to a student programmer; sometimes a secretary will also play an intermediate role, entering much of the program directly from the flowchart.

The following segment of code corresponds to the flowchart segment. The macro calls are apparent; most of the macros have English language names, and their functions in most cases are obvious, particularly if you refer to the flowchart. Thus, WRITE produces terminal output, DELAY adds timed delays, SKIP skips lines on the screen, RESET makes a variable zero, CURSOR turns on the crosshairs for graphic input, BUMP increments a variable. Some of the macros are particularly concerned with graphics; WINDOW determines the active area of the screen, SCALE establishes user coordinates for this area, CURVE draws a curve, ARROW draws an arrow.

BEST COPY AVAILABLE

```
29.000           NEWPAGE
30.000           SKIP      2
31.000           TO        SKIPIT1,(BAD,EQ,0)
32.000           TO        MEDIUM,(BAD,LT,3)
33.000           WRITE     'YOU''VE HAD QUITE A BIT OF TROUBLE. LET''S SEE'
34.000           WRITE     'IF IT HAS CLEARED UP'
35.000           SKIP      2
36.000 MEDIUM    WRITE     'GIVEN CHARGES Q1 AND Q2 SEPARATED BY A DISTANCE'
37.000           WRITE     '''R'', THE FORCE IS PROPORTIONAL TO          HHHHHHHH'
38.000 QUES2     INBELL
39.000           NOBLANK
40.000           SUBALL    '**',' '
41.000           DELETEALL '*'
42.000           SUBALL    'RR','R*'
43.000           IF        'R*R',SWITCH
44.000           SUBALL    'R*','R 2'
45.000           IF        ('Q1Q2/R 2','Q2Q1/R 2'),GREAT
46.000 SWITCH    SWITCH    BAD,(STRK1,STRK2,STRK3,STRK3)
47.000 STRK1     WRITE 'YOU HAVE ALL THE COMPONENTS, THE FORCE IS PROPORTIONAL'
48.000           WRITE     'TO EACH CHARGE AND INVERSELY PROPORTIONAL TO'
49.000           WRITE     'THE SQUARE OF THE SEPARATION'
50.000           DELAY     2
51.000           BUT       'TRY ONCE MORE TO'
52.000           WRITE     'TO TELL US WHAT EXPRESSION THE FORCE IS'
53.000           WRITE     'PROPORTIONAL TO '
54.000           BUMP      BAD
55.000           TO        QUES2
56.000 STRK2     BUMP      BAD
57.000           WRITE     'THE FORCE IS PROPORTIONAL TO Q1Q2/R 2',SKIPIT
58.000 STRK3     WRITE     'STILL TROUBLE. PERHAPS YOU SHOULD CONSULT A '
59.000           WRITE     'PHYSICS FACULTY MEMBER, OR ANOTHER STUDENT.'
60.000           BUMP      BAD
61.000           TO        SKIPIT
62.000 GREAT     TO        FINE4,(BAD,LT,3)
63.000           WRITE     'YOU SEEM TO HAVE THE IDEA NOW.',SKIPIT
64.000 FINE4     WRITE     'FINE'
65.000 SKIPIT    SKIP
66.000           DELAY     2
67.000           NEWPAGE
68.000           SKIP      5
69.000 SKIPIT1   WRITE     'THE PROPORTIONALITY CONSTANT DEPENDS ON THE UNITS'
70.000           WRITE     'CHOSEN. IN ONE COMMON SET OF UNITS, SI (OR MKS),'
71.000           DELAY     1
72.000           SKIP
73.000           WRITE     '              1      Q1 Q2'
74.000           WRITE     '      F = -------  -------'
75.000           WRITE     '          4 PI EO    R 2'
76.000           SKIP
77.000           DELAY     2
78.000           WRITE     'WHERE PI=3.14159...., CHARGE IS IN COULOMBS,'
79.000           WRITE     'DISTANCES IS IN METERS, AND EO, CALLED'
80.000           WRITE     'EPSILON-NAUGHT, IS 9.0*10 9 NEWTONS/COULOMBS.'
81.000           RESET     BAD2
82.000           SKIP
83.000           DELAY     2
```

BEST COPY AVAILABLE

```
84.000          NEWPAGE
85.000          WRITE    'NEXT ON OUR AGENDA IS THE DIRECTION OF THE FORCE.'
86.000          DELAY    1
87.000          WRITE    'HERE''S A POSITIVELY CHARGED PARTICLE, ''A'''
88.000          DELAY    2
89.000          WINDOW   (FS'2.',FS'.4'),(FS'6.',FS'2.9'),BOX
90.000          SCALE    (FS'0.',FS'32.'),(FS'0.',FS'20.')
91.000          FORTRAN  PLACE,(Z8,Z9)
92.000          CURVE    (X,Y,30)
93.000          SETPOINT (FS'2.75',FS'1.175')
94.000          BUT      'A'
95.000          DELAY    2
96.000          WRITE    'AND ANOTHER, ''B'', ALSO POSITIVE'
97.000          DELAY    1
98.000          FORTRAN  PLACE,(Z10,Z11)
99.000          CURVE    (X,Y,30)
00.000          SETPOINT (FS'4.95',FS'2.375')
01.000          BUT      'B'
02.000          DELAY    3
03.000          SKIP
04.000          WRITE    'THE FORCE ON ''B'' IS A VECTOR QUANTITY.'
05.000          SKIP
06.000          WRITE    'PLEASE INDICATE ITS DIRECTION BY PLACING'
07.000          WRITE    'THE CROSSHAIRS IN A PROPER DIRECTION'
08.000          WRITE    'RELATIVE TO THE CENTER OF THE CHARGE ''B'''
09.000          WRITE    'AT ''B''.'
10.000          DELAY    3
11.000          BUT       ' POINT WITH THE CROSSHAIRS TO THE TIP'
12.000          WRITE    'OF THE ARROW AND THEN TYPE A ''P''.'
13.000 CURSOR   CURSOR   A,B
14.000          FORTRAN  ACCEPT,(A,B,C,D,J)
15.000          LW,1     J
16.000 GADR     B        GADR,1
17.000          B        RIGHT
18.000          B        CLUMSY
19.000          B        REVERSE
20.000          TO       CONTIN,(BAD2,GT,0)
21.000          BUMP     BAD2
22.000          WRITE    'THAT DOESN''T SEEM RIGHT, TRY AGAIN',CURSOR
23.000 RIGHT    WRITE    'LOOKS GOOD'
24.000          RESET    BAD2
25.000          TO       CONTIN
26.000 REVERSE  WRITE    'NO    THE FORCE IS '
27.000          PRINT    (REPEAT,5,'REPULSIVE')
28.000          BUT      ' NOT ATTRACTIVE.'
29.000          WRITE    'TRY AGAIN',CURSOR
30.000 CLUMSY   WRITE    'YOU''RE A LITTLE CARELESS, BUT I ASSUME YOU '
31.000          WRITE    'MEAN AN ARROW IN THIS DIRECTION.'
32.000          RESET    BAD2
33.000 CONTIN   DELAY    2
34.000          ARROW    (FS'24.',FS'14.4'),(FS'26.634',FS'15.837')
35.000          WRITE    'THE FORCE IS ALONG THE LINE JOINING THE'
36.000          WRITE    'TWO PARTICLES',SECT2
37.000 A        DATA     0
38.000 B        DATA     0
```

Note that FORTRAN is also used in the preparation of dialogs. FORTRAN subroutines are callable from the macro-based portions of the programs, and some of the facilities of the macros are callable from FORTRAN. The primary use of FORTRAN is for calculation, the area in which it is particularly strong. We can make extensive use, too, of the available FORTRAN subroutine libraries. For example, a number of our Irvine dialogs include programs from the Scientific Subroutine Package, a common collection available on most sizable machines.

We believe that this combination of macros plus FORTRAN allows us the advantages mentioned above, without our developing a special purpose language. To some extent the dialog commands are a special purpose language (the term "language" does not have precise meaning), but the emphasis is very different from what is usual in language construction. The primary difference is that it is a growing set of facilities, growing by pedagogical demand rather than an entity defined in advance. Further, we do not assume that the teacher will be familiar with these language facilities.

Future

First, the most obvious comment to make about the future is that the Irvine dialog facilities are always developing. We are at the moment, for example, writing an elaborate new set of macros having to do with symbol manipulation and parsing of complicated expressions involving complex numbers and operators. These macros like others that have been developed are being coded because they are necessary for the dialogs currently under development at Irvine.

Since we are working in a general purpose environment, we can use other software facilities in the machine besides those already employed. The most promising, particularly for some of our new applications which combine computer management along with computer dialogs, are the full database facilities available. These, like any other software in the machine, can be called within macro-based code, and so programs can use this database facility in rapidly developing new facilities in this area. We are planning such development at this time, with emphasis on course management.

If we look a little further into the future, we can see other exciting possibilities. While we are currently working in an assembly language, there is increasing interest in system development work within a systems programming language. While valid questions can be raised about the efficiency of code produced by such languages, and the practicality of this approach, it is an interesting one to explore for the future. Hence, one might see the possibility of developing facilities like those described for the Irvine dialogs within a high level system programming language, perhaps as a structural addition to the language. Other languages, as indicated, could still be used. If a single system programming language gains general acceptance, then this might ease the process of transferability; but this possibility does not look likely.

A more likely development for the future of educational computing, perhaps, is the rise of widespread use of stand-alone systems, displays having within them most of the processing

capabilities that they will usually need. These systems, unlike timesharing systems, will be used by individual students. So if a particular machine breaks down, only that student will be affected, and not the hundreds of students who might be affected in a timesharing environment. There are also great advantages with this type of machine with regard to graphic display capabilities.

If such machines become common in the educational evironment in a period within five to ten years from now, the task of preparing and testing the interactive software and the dialogs will probably still go on in the larger timesharing systems that we are already developing. Hence, software systems such as I described may still play a role in generating the programs for the small machines.

Conclusions

I presented here an alternate to the development of a special purpose language for student computer dialogs and examined possibilities for the future. We believe a system of this kind is effective for generating highly interactive material valuable for students in the learning process.

The work described here is supported by the National Science Foundation and the University of California. A bibliography and documentation of the underlying software and of individual dialogs are available on request.